UNIVERSIDAD NACIONAL EXPERIMENTAL DEL TÁCHIRA
DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA
SISTEMAS OPERATIVOS COD. 0425603T
LABORATORIO 3 – HILOS EN POSIX

Objetivo de la práctica

El estudiante aprenderá a desarrollar aplicaciones multi-plataforma (específicamente múltiples sistemas operativos), utilizando hilos del sistema a través de la API del S.O. Para acceder a variables globales y modificar su valor de manera segura usando semáforos.

Conocimientos previos

- Desarrollo de aplicaciones en C/C++
- Concepto de Hilo o Hebra, Semáforos, Sección Crítica.
- Uso del API de Windows y/o POSIX.

EJEMPLO DE HILOS CON SEMÁFOROS EN GNU/LINUX:

A continuación se muestra un ejemplo de un programa que utiliza hilos de la API POSIX, ejecútelo y observa la salida correspondiente. Para su ejecución se debe compilar el archivo .c de la siguiente manera:

gcc –lpthread programa.c **–o** programa.out

```
#include <pthread.h> //<-- liberia de POSIX para crear Hilos
#include <stdio.h>
#include <stdlib.h>

// Librerías globales que comparten los hilos
int sum = 0, numero;

//Semoforo para controlar la sincronización de hilos
```

```
pthread_mutex_t mutex_acceso;

void *runner(void *param); // La hebra
int main(int argc, char *argv[]) {
    pthread_t tid1, tid2; // identificador de las hebras
    pthread_attr_t attr; // conjunto de atributos de la hebra

    if (argc != 2) {
        fprintf(stderr, "Modo de uso : a.out valor entero\n");
        return -1;
    }

   numero = atoi(argv[1]);

    if (numero < 0) {
        fprintf(stderr, "%d debe ser >=0\n", numero);
        return -1;
    }

    // Obtener los atributos predeterminados
    pthread_attr_init(&attr);

    //Inicializa el semaforo
    pthread_mutex_init(&mutex_acceso, NULL);

    //Crear la hebra
    pthread_create(&tid1, &attr, runner, &numero);
    pthread_create(&tid2, &attr, runner, &numero);
```

```
    //Espera a que las hebras termine
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);

    sum /=2 ; //Arreglar para que cada hilo funcione correctamente y evitar esta linea
    printf("sum= %d\n", sum);
}
/*La hebra inicia su ejecución en esta función*/
void *runner(void *param) {
    int i, numero;
   // numero = int param; en 32 bits
    numero = *((int *) param); //64 bits
    pthread_mutex_lock(&mutex_acceso);

    for (i = 1; i <= numero; i++)
        sum += i;

    pthread_mutex_unlock(&mutex_acceso);
    pthread_exit(0);
}
```

ASIGNACIÓN

Crackeador de MD5

Una compañía de seguridad informática está buscando futuros hackers para desarrollar un conjunto de herramientas de auditoría de seguridad. Pero para ser parte del equipo, solicitan el desarrollo de una herramienta para crackear contraseñas cifradas bajo el algoritmo MD5.

Esta herramienta recibe un archivo de entrada llamado *users.txt*, donde cada línea representa una cuenta de usuario y la información esta almacenada bajo el siguiente formato: **usuario::contraseña_cifrada**

Su programa debe leer cada línea del archivo de entrada, crackear la contraseña y mostrar por pantalla el nombre del usuario junto a la contraseña, separado por una tabulación. Debido a la gran cantidad de cálculos matemático que requiere para crackear las contraseñas, debe resolver el problema utilizando hilos para aprovechar al máximo el hardware.

Para facilitar el desarrollo, se han agregado las siguientes restricciones:

- Las contraseñas poseen un tamaño máximo de 4 caracteres, compuesto de letras minúsculas o mayúsculas ([a-zA-z]).
- La cantidad máxima de usuarios a procesar es 50 y la cantidad máxima de hilos a ejecutar simultáneamente son 6.
- El programa debe permitir seleccionar la ubicación del archivo y la cantidad de hilos a usar. Utilice parámetros de entrada para usar estas opciones.
- Los formatos de salida deben ser respetados pues serán utilizados como datos de entrada para otro programa.

Ejemplo de entrada

miguel::47bce5c74f589f4867dbd57e9ca9f808

maria::cc19ea8ef3c58bca707a6386ad504bd4

hchavez2021::99c5d9016c7638d054780f65970d4aa7

zerocool::942af6caa509180737efb1b1968f3612

Ejemplo del formato del archivo de salida

miguel aaa

maria CoFFee

hchavez2021 FiDEl

zerocool hAcK

Limitaciones:

- El parámetro para la cantidad de hilos/hebras se pasa como número al momento de ejecutar el programa, nunca debe ser solicitado al usuario.
- Los hilos deben ejecutarse en paralelo y el hilo principal esperar a que terminen.
- El programa **debe utilizar semáforos o mecanismos de exclusión mutua** para evitar mala sobre-escritura del archivo de salida.
- No se permite el uso de librerías multi-plataforma de hilos, deben crearse manualmente las nativas que proporciona el API del sistema operativo.
- El código fuente del programa debe ser uno solo, no se permite crear dos aplicaciones independientes para cada S.O.